FailSafe™ Architecture for SGI® InfiniteStorage

007–4650–001

CONTRIBUTORS

Written by Lori Johnson

Illustrated by Chrystie Danzer

Engineering contributions by Wesley Smith, Paddy Sreenivasan

COPYRIGHT

© 2003 Silicon Graphics, Inc. All rights reserved; provided portions may be copyright in third parties, as indicated elsewhere herein. No permission is granted to copy, distribute, or create derivative works from the contents of this electronic documentation in any manner, in whole or in part, without the prior written permission of Silicon Graphics, Inc.

LIMITED RIGHTS LEGEND

The electronic (software) version of this document was developed at private expense; if acquired under an agreement with the USA government or any contractor thereto, it is acquired as "commercial computer software" subject to the provisions of its applicable license agreement, as specified in (a) 48 CFR 12.212 of the FAR; or, if acquired for Department of Defense units, (b) 48 CFR 227-7202 of the DoD FAR Supplement; or sections succeeding thereto. Contractor/manufacturer is Silicon Graphics, Inc., 1600 Amphitheatre Pkwy 2E, Mountain View, CA 94043-1351.

TRADEMARKS AND ATTRIBUTIONS

IRIS, IRIX, Silicon Graphics, SGI, and the SGI logo are registered trademarks and CXFS, FailSafe, and IRIS FailSafe are trademarks of Silicon Graphics, Inc., in the United States and/or other countries worldwide.

Netscape and Netscape FastTrack Server are trademarks of Netscape Communications Corporation. Oracle is a registered trademark of Oracle Corporation. UNIX is a registered trademark of The Open Group in the United States and other countries.

Cover design by Sarah Bolles, Sarah Bolles Design, and Dany Galgani, SGI Technical Publications.

Record of Revision

Version	Description
001	August 2003 Original publication

Contents

Figures

Tables

Introduction

FailSafe provides a general facility for providing highly available services. It is supported on IRIX systems.

If a failure occurs, a different node in the cluster restarts the highly available services of the failed node. To clients, the services on the replacement node are indistinguishable from the original services before failure occurred. It appears as if the original node has crashed and rebooted quickly. The clients notice only a brief interruption in the highly available service.

In a FailSafe environment, nodes can serve as backup systems for other nodes. Unlike the backup resources in a fault-tolerant system, which serve purely as redundant hardware for backup in case of failure, the resources of each node in a highly available system can be used during normal operation to run other applications that are not necessarily highly available services. All highly available services are owned by one node in the cluster at a time.

Highly available services are monitored by the FailSafe software. If a failure is detected on any of these components, a failover process is initiated. Using FailSafe, you can define a failover policy to establish which node will take over the services under what conditions. This process consists of resetting the failed node (to ensure data consistency), performing recovery procedures required by the failed over services, and quickly restarting the services on the node that will take them over.

This paper discusses the following aspects of FailSafe operations:

Software Layers

A FailSafe system has the following software layers:

- Plug-ins, which create highly available services. Table 2-1 on page 4, and Table 2-2 on page 4, show the provided and optional FailSafe plug-ins and their associated resource types.

  See the release notes for information about the specific releases of these products that are supported.

  Note: The Samba `interfaces` parameter allows Samba to support multiple IP interfaces. It takes the following format, where *IP* **must be** a dotted decimal IP address and *netmask* **must be** a dotted decimal netmask such as 255.255.255.0:

  `interfaces =` *IP1/netmask1 IP2/netmask2*

  If the application you want is not available, you can hire the SGI Professional Services group to develop the required software, or you can use the *FailSafe Programmer's Guide for SGI Infinite Storage* to write the software yourself.

- FailSafe base, which includes the ability to define resource groups and failover policies.
- Cluster services, which lets you define clusters, resources, and resource types (this consists of the `cluster_services` installation package)
- Cluster software infrastructure, which lets you do the following:
  - Perform node logging
  - Administer the cluster
  - Define nodes

  The cluster software infrastructure consists of the `cluster_admin` and `cluster_control` subsystems.

Table 2-1 Provided Plug-Ins

Provided Plug-In	Resource Type
CXFS file system	`CXFS`
IP addresses	`IP_address`
MAC addresses	`MAC_address`
XFS filesystems	`filesystem`
XLV logical volumes	`volume`
XVM volume manager	`XVM`

Table 2-2 Optional Plug-Ins

Optional Plug-In	Resource Type
FailSafe for DMF	`DMF`
FailSafe for NFS	`NFS` and `statd_unlimited`
FailSafe for Informix	`INFORMIX_DB`
FailSafe for Oracle	`Oracle_DB`
FailSafe for Samba	`Samba`
FailSafe for TMF	`TMF`
FailSafe for Web (Netscape)	`Netscape_web`

Figure 2-1 shows a graphic representation of these layers. The cluster services and cluster software infrastructure layers are shared with CXFS. Table 2-3 on page 6, describes the contents of the `/usr/cluster/bin` directory. For more information about CXFS, see the *CXFS Administration Guide for SGI Infinite Storage*.

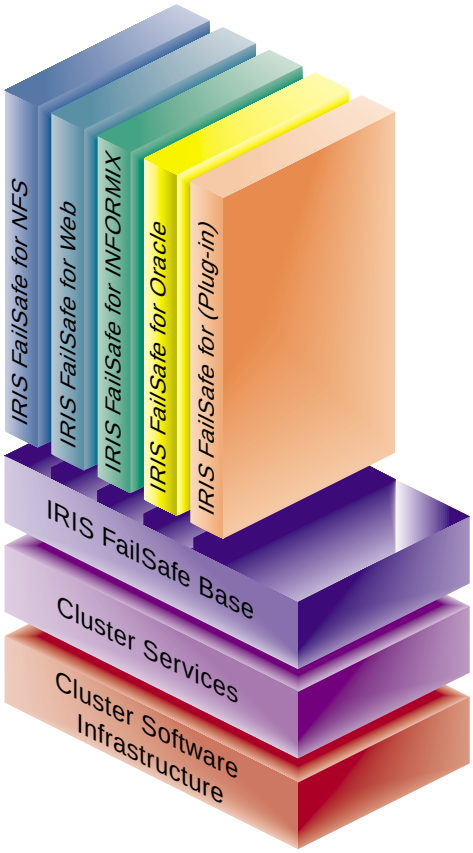

Figure 2-1 Software Layers

Table 2-3 Contents of `/usr/cluster/bin`

Layer	Subsystem	Process	Description
Plug-ins	`failsafe_informix` `failsafe2_oracle`	`ha_ifmx2`	FailSafe database agents. Each database agent monitors all instances of one type of database.
FailSafe Base	`failsafe2`	`ha_fsd`	FailSafe daemon. Provides basic component of the FailSafe software.
Cluster services (high-availability processes)	`cluster_services`	`ha_cmsd`	The FailSafe membership daemon. Provides the list of nodes, called *FailSafe membership*, available to the cluster.
		`ha_gcd`	Group membership daemon. Provides group membership and reliable communication services in the presence of failures to FailSafe processes.
		`ha_srmd`	System resource manager daemon. Manages resources, resource groups, and resource types. Executes action scripts for resources.
		`ha_ifd`	Interface agent daemon. Monitors the local node's network interfaces.
Cluster software infrastructure (cluster administrative processes)	`cluster_admin`	`cad`	Cluster administration daemon. Provides administration services.
	`cluster_control`	`crsd`	Node control daemon. Monitors the serial connection to other nodes. Has the ability to reset other nodes.
		`cmond`	Daemon that manages all other daemons. This process starts other processes in all nodes in the cluster and restarts them on failures.
		`fs2d`	Manages the cluster database and keeps each copy in sync on all nodes in the pool.

Communication Paths

This chapter discusses the communication paths in a cluster running FailSafe and in a cluster running both FailSafe and CXFS (known as a *coexecution cluster*).

Communication Paths in a FailSafe Cluster

The following figures show communication paths in FailSafe.

Note: The following figures do not represent the `cmond` cluster manager daemon. The purpose of this daemon is to keep the other daemons running.

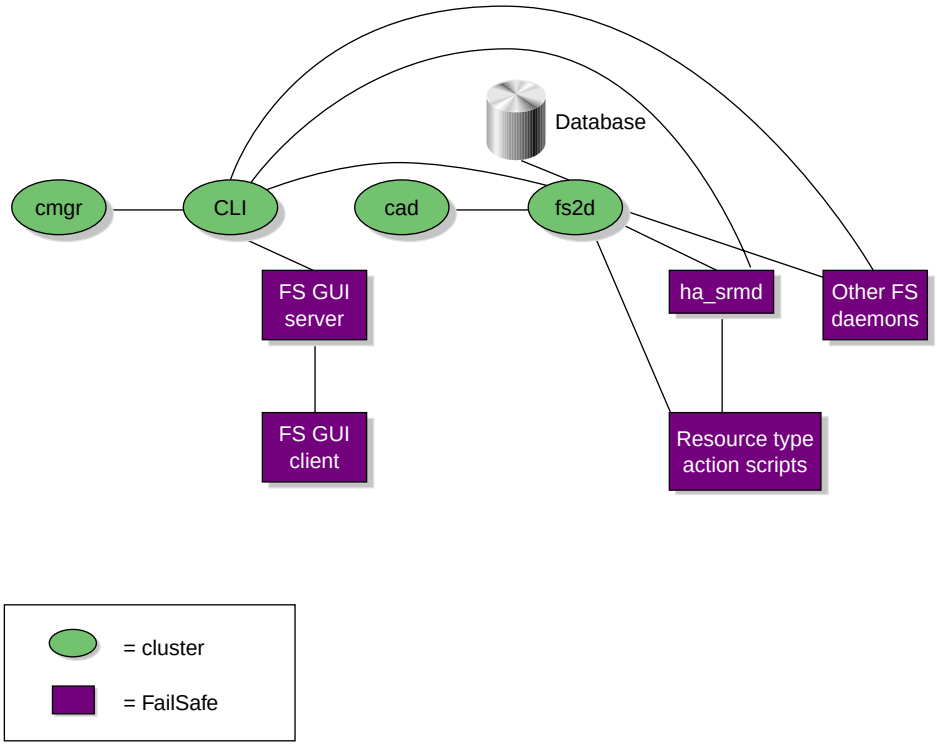

Figure 3-1 Administrative Communication within One Node

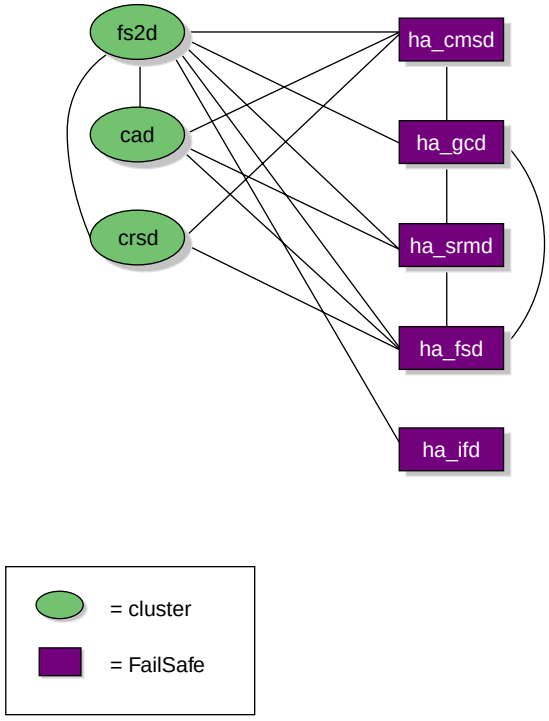

Figure 3-2 Daemon Communication within One Node

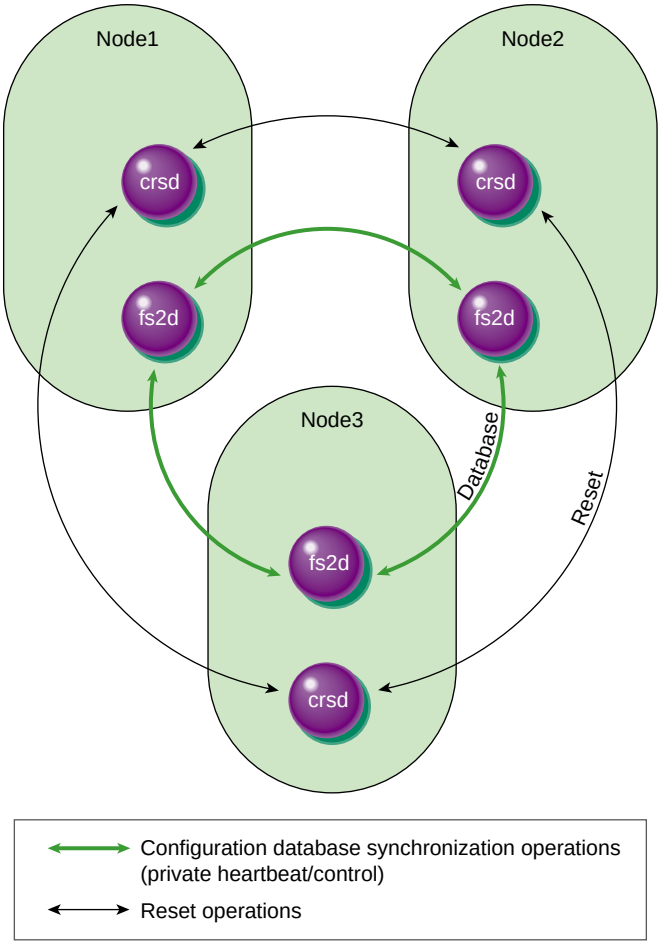

Figure 3-3 Communication between Nodes in the Pool

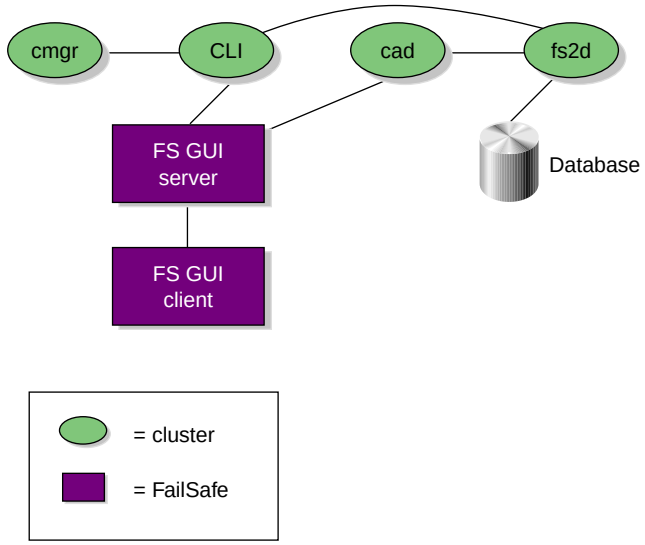

Figure 3-4 Communication for a Node Not in the Cluster

Communication Paths in a Coexecution Cluster

The following figures show the communication paths within one node in a coexecution cluster.

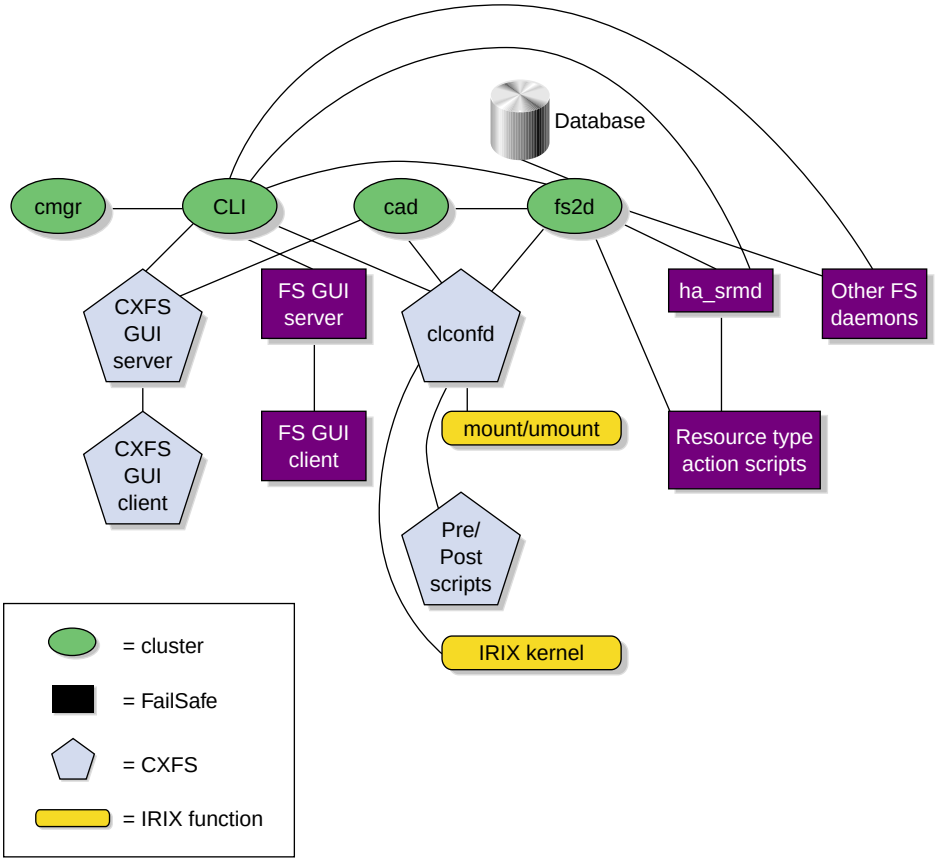

Figure 3-5 Administrative Communication within One Node under Coexecution

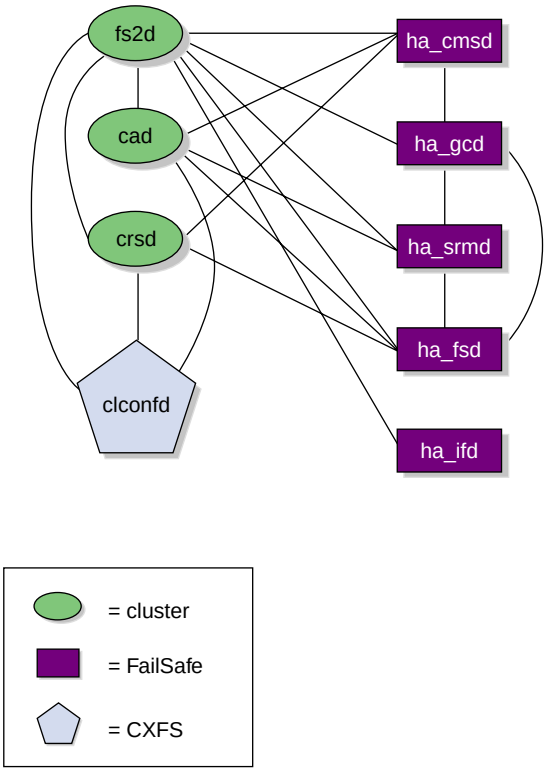

Figure 3-6 Daemon Communication within One Node under Coexecution

Cluster Database Components

The cluster database is a key component of FailSafe software. It contains all information about the following:

- Resources
- Resource types
- Resource groups
- Failover policies
- Nodes
- Clusters

The cluster database daemon (`fs2d`) maintains identical databases on each node in the cluster.

The following table shows the contents of the `/var/cluster/ha` directory.

Table 4-1 Contents of the `/var/cluster/ha` directory

Directory or File	Purpose
`comm/`	Directory that contains files that communicate between various daemons. FailSafe processes create temporary files in this directory. FailSafe interprocess communication will fail if there is not sufficient disk space for this directory (approximately 2–3 MB) in the root filesystem on every node in a FailSafe cluster.
`common_scripts/`	Directory that contains the script library (the common functions that may be used in action scripts).
`log/`	Directory that contains the logs of all scripts and daemons executed by FailSafe. The outputs and errors from the commands within the scripts are logged in the `script_`*Nodename* file.

Directory or File	Purpose
`policies/`	Directory that contains the failover scripts used for resource groups.
`resource_types/template`	Directory that contains the template action scripts.
`resource_types/`*RTname*	Directory that contains the action scripts for the *RTname* resource type. For example, `/var/cluster/ha/resource_types/filesystem`.
`resource_types/`*RTname*`/exclusive`	Script that verifies that a resource of this resource type is not already running.
`resource_types/`*RTname*`/monitor`	Script that monitors a resource of this resource type.
`resource_types/`*RTname*`/restart`	Script that restarts a resource of this resource type on the same node after a monitoring failure.
`resource_types/`*RTname*`/start`	Script that starts a resource of this resource type.
`resource_types/`*RTname*`/stop`	Script that stops a resource of this resource type.

Memberships and Quorums

The nodes in a cluster must act together to provide a service. To act in a coordinated fashion, each node must know about all the other nodes currently active and providing the service. The set of nodes that are currently working together to provide a service is called a membership. Cluster activity is coordinated by a configuration database that is replicated or at least accessible on all nodes in the cluster. The cluster software sends heartbeat messages between the nodes to indicate that a node is up and running. Heartbeat messages for each membership type are exchanged via a private network so that each node can verify each membership.

Nodes within the cluster must have the correct memberships in order to provide services. This appendix discusses the different types of membership and the effect they have on the operation of your cluster.

Nodes might not be able to communicate for reasons such as the following:

- They are down.
- The communication daemons have failed or have been turned off.
- Software has not been configured, or has been misconfigured.
- The network is misconfigured (in this case, some heartbeat messages may fail while others succeed).
- The network router or cable fails (in this case, all heartbeat messages will fail).

Nodes that cannot communicate must be excluded from the membership because the other nodes will not be able to verify their status.

It is critical that only one membership of each type exist at any one time, as confusion and corruption will result if two sets of nodes operate simultaneously but independently. There is a risk of this happening whenever a segmentation of the private network occurs, or any other network problem occurs that causes the nodes eligible for membership to be divided into two or more sets, where the nodes in each set can communicate with themselves, but not with nodes outside of the set. Thus, in order to form a membership, the nodes must have a quorum, the minimum number of nodes required to form a membership. The quorum is typically set at half the total eligible members.

For example, consider the case of six nodes eligible for a membership:

- If all six nodes can communicate with each other, they will form a membership of six and begin offering the membership's services.
- If a network segmentation occurs that causes four nodes to be in one set and two in another set, the two-node set will try to form its own membership but will be unable to do so because it does not have enough nodes to form a quorum; these nodes will therefore stop offering services. The four-node set will be able to form a new membership of four nodes and will continue to offer the membership's services.
- If a network segmentation occurs that divides the nodes into three sets of two nodes each, no set will be able to form a membership because none contains enough nodes to form a quorum. In this case, the membership services will be unavailable; this situation is unavoidable, as each set of two nodes thinks that the four other nodes may have formed a quorum, and so no set may safely offer the membership's services.
- If a network segmentation occurs that divides the nodes into two sets of three, then both could have a quorum, which could cause problems. To prevent this situation from occurring, some memberships may require a majority (>50%) of nodes or a tiebreaker node to form or maintain a membership. Tiebreaker nodes are used when exactly half the nodes can communicate with each other.

The following sections provide more information about the specific requirements for membership.

Note: Because the nodes are unable to distinguish between a network segmentation and the failure of one or more nodes, the quorum must always be met, regardless of whether a partition has actually occurred or not.

Membership Types

There are three types of membership:

- "Cluster Database Membership and Quorum"
- "FailSafe Membership, Quorum, and Tiebreaker" on page 20
- "CXFS Kernel Membership, Quorum, and Tiebreaker" on page 21

Each provides a different service using a different heartbeat. Nodes are usually part of more than one membership.

Cluster Database Membership and Quorum

The nodes that are part of the the cluster database membership (also known as `fs2d` *membership*) work together to coordinate configuration changes to the cluster database:

- The *potential* cluster database membership is all of the administration nodes (installed with `cluster_admin` and running `fs2d`) that are defined using the GUI or the `cmgr` command as nodes in the pool. (CXFS client-only nodes are not eligible for cluster database membership.)
- The *actual* membership is the subset of eligible nodes that are up and running and accessible to each other, as determined by heartbeats on the private network. If the primary private network is unavailable, the cluster database heartbeat will fail over to the next available heartbeat network defined for the node, if any (CXFS nodes are limited to a single heartbeat network).

The cluster database heartbeat messages use remote procedure calls (RPCs). Heartbeats are performed among all nodes in the pool. You cannot change the heartbeat timeout or interval.

If a node loses its cluster database membership, the cluster database write-operations from the node will fail; therefore, FailSafe and CXFS configuration changes cannot be made from that node.

The *cluster database membership quorum* ensures atomic write-operations to the cluster database that `fs2d` replicates in all administration nodes in the pool.

The quorum allows an initial membership to be formed when a majority (>50%) of the eligible members are present. If there is a difference in the membership log between members, the cluster database tiebreaker node is used to determine which database is replicated. (See "Cluster Database Membership Logs" on page 23.) The tiebreaker node is always the administration node in the membership with the lowest node ID; you cannot reconfigure the tiebreaker for cluster database membership.

When the quorum is lost, the cluster database cannot be updated. This means that FailSafe and CXFS configuration changes cannot be made; although FailSafe and CXFS may continue to run, the loss of the cluster database quorum usually results in the loss of quorum for FailSafe and/or CXFS, because the nodes that drop from the cluster database membership will probably also drop from other memberships.

FailSafe Membership, Quorum, and Tiebreaker

The nodes that are part of the FailSafe membership provide highly available (HA) resources for the cluster:

- The *potential* FailSafe membership is the set of all FailSafe nodes that are defined in the cluster and on which HA services have been enabled. Nodes are enabled when HA services are started. The enabled status is stored in the cluster database; if an enabled node goes down, its status will remain enabled to indicate that it is supposed to be in the membership.
- The *actual* membership consists of the eligible nodes whose state is known and that are communicating with other FailSafe nodes using heartbeat and control networks. If the primary private network is unavailable, the FailSafe heartbeat will fail over to the next available heartbeat network defined for the node.

Stopping HA services on a node (deactivating the node) is equivalent to removing a node from FailSafe cluster. FailSafe membership does not include deactivated nodes in membership calculation.

The FailSafe heartbeat uses user datagram protocol (UDP). Heartbeats are performed among all FailSafe-enabled nodes in the cluster. You can change the FailSafe heartbeat timing with the GUI **Set FailSafe HA Parameters** task or the `cmgr` command `modify ha_parameters` (the `node_timeout` parameter is the heartbeat timeout and the heartbeat is the heartbeat interval).

If a node loses its FailSafe membership, FailSafe will fail over its HA resources to another node in the cluster.

The *FailSafe membership quorum* ensures that a FailSafe resource is available only on one node in the cluster. The quorum requires that the state of a majority (>50%) of eligible nodes to be known and that half (50%) of the eligible nodes be present to form or maintain membership.

If a network partition results in a tied membership, in which there are two sets of nodes (each consisting of 50% of the potential FailSafe membership), then a node from the set containing the *FailSafe tiebreaker node* will attempt to perform a serial hardware reset on a node in the other set. *Serial hardware reset* is the failure action that performs a system reset via a serial line connected to the system controller.

If the node can verify that the other node was reset, then the membership will continue on the set with the tiebreaker. However, containing the tiebreaker is not a guarantee of membership; for more information, see the *FailSafe Administrator's Guide*

for SGI InfiniteStorage. The default FailSafe tiebreaker is the node with the lowest node ID in the cluster.

When FailSafe membership quorum is lost, the resources will continue to run but they are no longer highly available.

CXFS Kernel Membership, Quorum, and Tiebreaker

The nodes that are part of the CXFS kernel membership can share CXFS filesystems:

- The *potential* CXFS kernel membership is the group of all CXFS nodes defined in the cluster and on which CXFS services have been enabled. Nodes are enabled when CXFS services are started. The enabled status is stored in the cluster database; if an enabled node goes down, its status will remain enabled to indicate that it is supposed to be in the membership.
- The *actual* membership consists of the eligible nodes on which CXFS services have been enabled and that are communicating with other nodes using the heartbeat/control network. CXFS supports only one private network, and that network is the only network used for CXFS kernel membership heartbeats (but remember that the CXFS nodes may use multiple networks for the cluster database membership heartbeats).

  Note: CXFS metadata also uses the private network. The multiple heartbeats on the private network therefore reduce the bandwidth available for CXFS metadata.

  During the boot process, a CXFS node applies for CXFS kernel membership. Once accepted, the node can actively share the filesystems in the cluster.

The CXFS heartbeat uses multicast. Heartbeats are performed among all CXFS-enabled nodes in the cluster.

If a node loses its CXFS kernel membership, it can no longer share CXFS filesystems.

The *CXFS kernel membership quorum* ensures that only one metadata server is writing the metadata portion of the CXFS filesystem over the storage area network:

- For the *initial* CXFS kernel membership quorum, a majority (>50%) of the server-capable administration nodes with CXFS services enabled must be available to form a membership. (*Server-capable administration* nodes are those that are installed with the `cluster_admin` product and are also defined with the GUI or `cmgr` as capable of serving metadata. Client administration nodes are those that

are installed with the `cluster_admin` product but are not defined as server-capable.)

Note: Client administration nodes and client-only nodes can be part of the CXFS kernel membership, but they are not considered when forming a CXFS kernel membership quorum. Only server-capable nodes are counted when forming the quorum.

- To *maintain* the existing CXFS kernel membership quorum requires at least half (50%) of the server-capable nodes that are eligible for membership. If CXFS kernel quorum is lost, the shared CXFS filesystems are no longer available.

If you do not use serial hardware reset or I/O fencing to prevent a problem node from accessing I/O devices, you should set a CXFS tiebreaker node to avoid multiple CXFS kernel memberships in the event of a network partition. In CXFS, there is no default tiebreaker. Any node in the cluster can be a CXFS tiebreaker node. You can set the tiebreaker node by using the GUI's **Set Tiebreaker Node** task or by using the `set tie_breaker` command in `cmgr`.

Note: Suppose you have a cluster with only two server-capable nodes with one of them being the CXFS tiebreaker node. If the tiebreaker node fails or if the administrator stops CXFS services on the tiebreaker node, the other node will not be able to maintain a membership and will do a forced shutdown of CXFS services, which unmounts all CXFS filesystems.

If I/O fencing or serial hardware reset is used, the quorum is maintained by whichever side wins the reset/fence race.

If a tiebreaker node is set and the network being used for heartbeat/control is divided in half, only the group that has the CXFS tiebreaker node will remain in the CXFS kernel membership. Nodes on any portion of the heartbeat/control network that are not in the group with the tiebreaker node will exit from the membership. Therefore, if the heartbeat/control network is cut in half, you will not have an active metadata server on each half of the heartbeat/control network trying to access the same CXFS metadata over the storage area network at the same time.

Note: A tiebreaker node must be configured individually for CXFS and for FailSafe. In a coexecution cluster, these could be different nodes.

Cluster Database Membership Logs

Each `fs2d` daemon keeps a *membership log* that contains a history of each database change (write transaction), along with a list of nodes that were part of the membership when the write transaction was performed. All nodes that are part of the cluster database membership will have identical membership logs.

When a node is defined in the database, it must obtain a current copy of the cluster database and the membership log from a node that is already in the cluster database membership. The method used to choose which node's database is replicated follows a hierarchy:

1. If the membership logs in the pool share a common transaction history, but one log does not have the most recent transactions and is therefore incomplete, the database from a node that has the complete log will be chosen to be replicated.
2. If there are two different sets of membership logs, the database from the set with the most number of nodes will be chosen.
3. If there are two different sets of membership logs, and each set has an equal number of nodes, then the set containing the node with the lowest node ID will be chosen.

To ensure that the complete transaction history is maintained, do not make configuration changes on two different administration nodes in the pool simultaneously. You should connect the CXFS or FailSafe GUI to (or run the `cmgr` command on) a single administration node in the pool when making changes. However, you can use any node in the pool when requesting status or configuration information.

The following figures describe potential scenarios using the hierarchies.

Figure 5-1 on page 25, shows:

- Time 1: An established pool of three administration nodes sharing heartbeats, with node IDs 1-3, represented by the node names N1-N3. The tiebreaker node is the node in the membership with the lowest node ID. Each successive database write is identified by a letter in the membership log.
- Time 2: A new node, N4, is defined using `cmgr` or the GUI connected to node N1. Node N4 (node ID = 4) joins the pool. Its membership log is empty.
- Time 3: Because N1/N2/N3 have identical membership logs, the database is replicated from one of them. In this case, N2 is randomly chosen.

- Time 4: All nodes in the pool have identical membership logs.
- Time 5: A network partition occurs that isolates N1. Therefore, N1 can no longer receive database updates. Configuration changes are made by connecting the GUI to N2 (or running `cmgr` on node N2); this results in updates to the membership logs in N2, N3, and N4, but not to N1 because it is isolated.
- Time 6: The partition is resolved and N1 is no longer isolated. Because N2/N3/N4 have identical membership logs, and share the beginning history with N1, the database is replicated from one of them. N4 is chosen at random.
- Time 7: All nodes in the pool have identical membership logs.

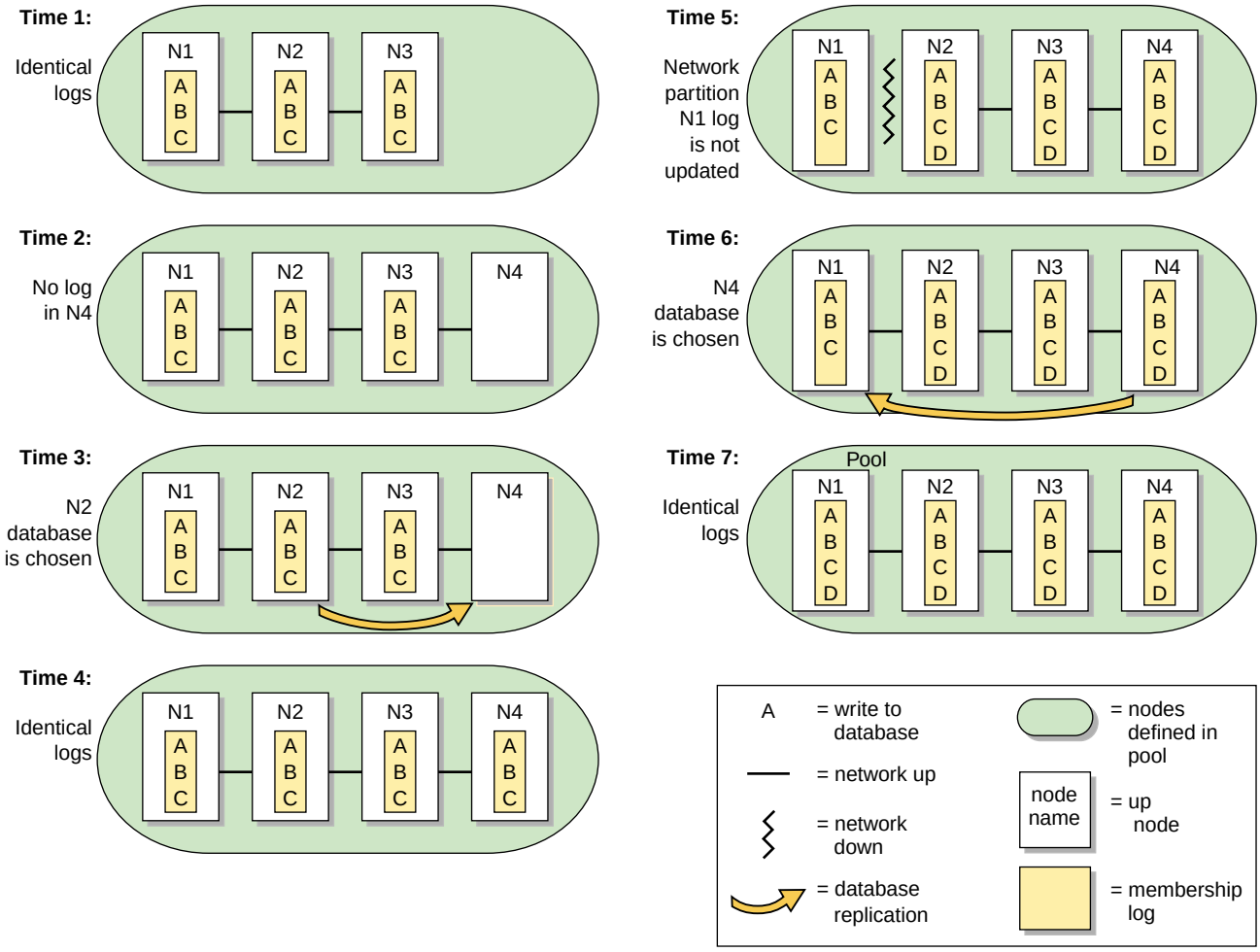

Figure 5-1 One Node is Out of Date: Most Recent Log is Replicated

Recall that a node can be in only one pool at a time. If there are two separate pools, and from a node in one pool you define one or more nodes that are already in the other pool, the result will be that nodes from one of the pools will move into the other pool. **This operation is not recommended**, and determining which nodes will move into which other pool can be difficult. Figure 5-2 on page 26 illustrates what to expect in this situation.

- Time 1: There are two pools that do not share membership log contents. One pool has two nodes (N1/N2), the other has three (N3/N4/N5).
- Time 2: N1 and N2 are defined as part of the second pool by running `cmgr` or connecting the GUI to node N3, N4, or N5. This results in a new pool with five nodes with different membership logs.
- Time 3: The database from the larger set of nodes is the one that must be replicated. N3 is chosen at random from the N3/N4/N5 set.
- Time 4: All nodes in the pool have identical membership logs.

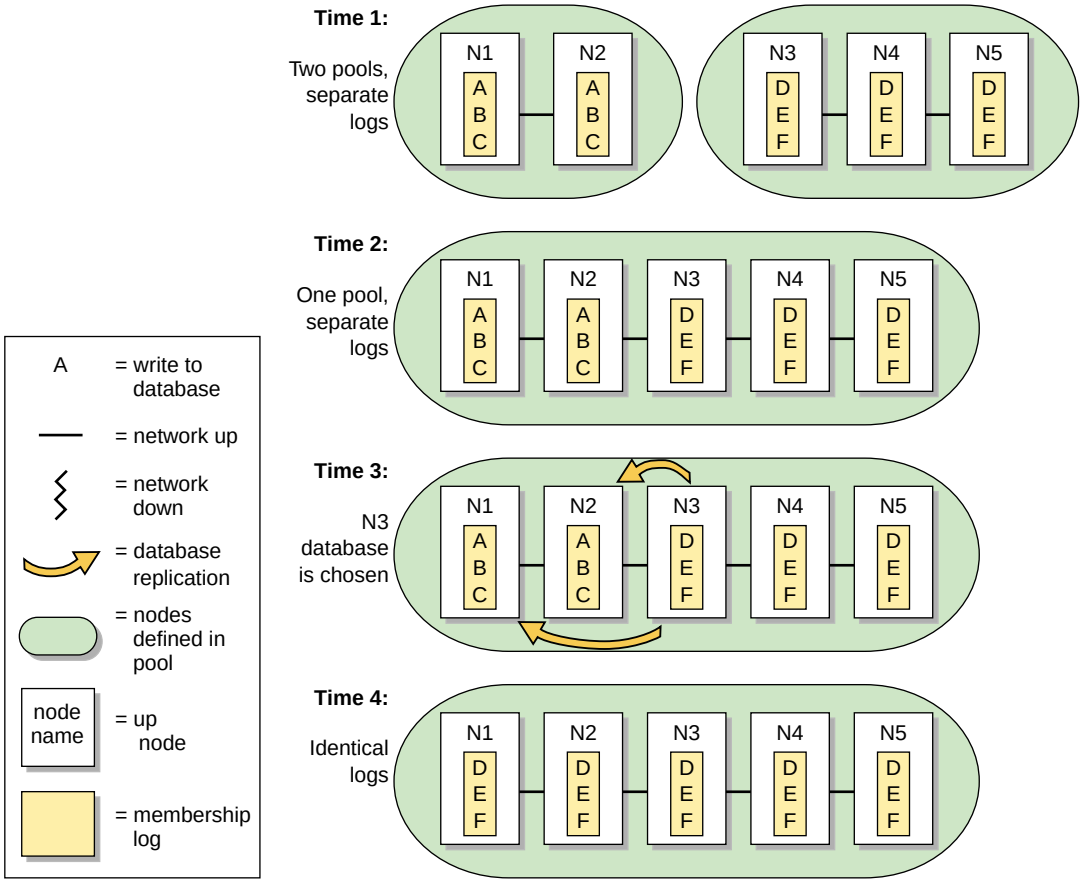

Figure 5-2 Unequally Sized Pools are Joined: Larger Pool Log is Replicated

Figure 5-3 on page 28, shows a similar situation in which two nodes are defined in two pools, but the pools are of equal size:

- Time 1: There are two pools that do not share membership log contents. Each pool has two nodes (N1/N2 in pool 1, and N3/N4 in pool 2).
- Time 2: N1 and N2 are defined as part of the second pool by connecting the GUI or running `cmgr` on node N3 or N4. This results in a new pool with four nodes with different membership logs.
- Time 3: Because each set has the same number of nodes, the tiebreaker node (the node with the lowest node ID in the membership) must be used to determine whose database will be chosen. Because node N1 is the lowest node ID (node ID=1), the database from N1 is chosen.
- Time 4: All nodes in the pool have identical membership logs.

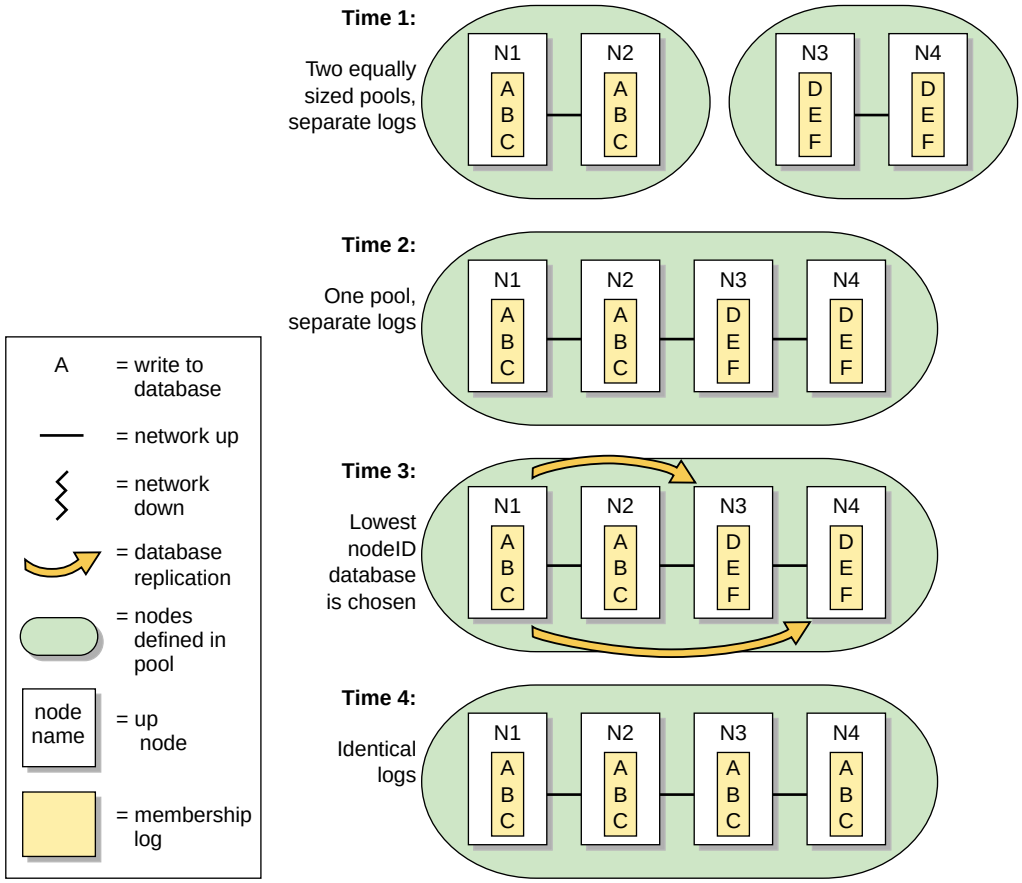

Figure 5-3 Equally Sized Pools are Joined: Lowest Node ID Log is Replicated

Quorum and Tiebreaker Examples

This section provides examples, of coexecution, FailSafe tiebreaker use, and cases when the FailSafe tiebreaker is not in the membership.

Coexecution Example

Figure 5-4 on page 30, shows an example of the different memberships in a cluster running CXFS and FailSafe. The pool contains 15 nodes (named N1 through N15). N1 has the lowest node ID number. There are CXFS client-only nodes running IRIX, Solaris, and Windows; only the nodes running IRIX are administration nodes containing the cluster database. The FailSafe nodes are those where HA services are enabled; each of these is an administration node.

- Cluster database membership:
  - Eligible: N1, N2, N3, N5, N9, and N10 (that is, all nodes containing the cluster database)
  - Actual: N1, N2, N3, and N10 (because N5 and N9 are down)
  - Quorum: N1, N2, N3, and N10 (>50% of eligible nodes)
- FailSafe membership:
  - Eligible: N1, N2, and N3 (that is, those nodes with HA services enabled and defined as part of the cluster)
  - Actual: N1, N2, N3
  - Quorum: N1, N2, N3 (>50% of eligible nodes)
- CXFS kernel membership:
  - Eligible: N1-N8 and N11-N15 (N9 and N10 are not defined as part of the cluster)
  - Actual: N1, N2, N3, N4, N6, and N11-N15 (because N5, N7, and N8 are down)
  - Quorum: N1, N2, N3 (>50% of server-capable eligible nodes)

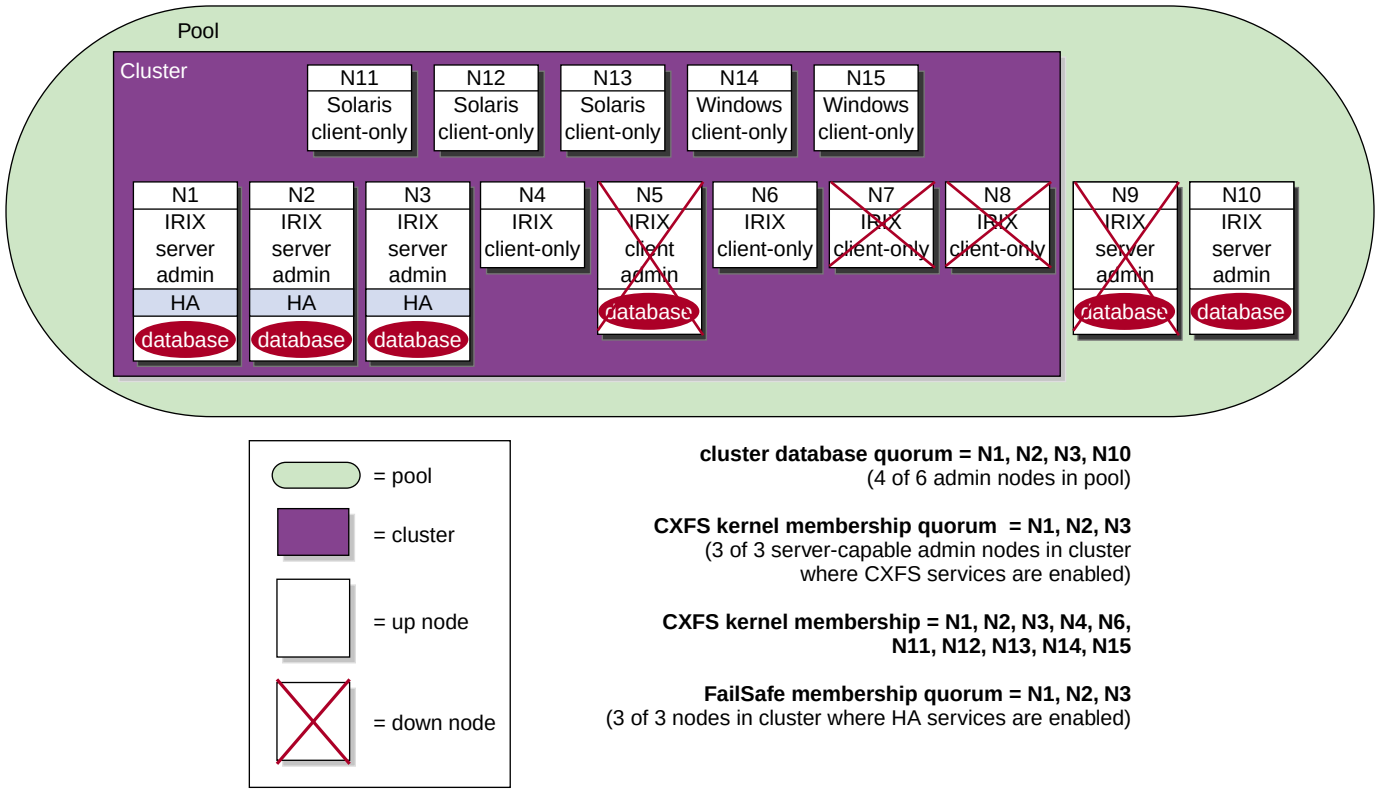

Figure 5-4 Example Memberships in a Coexecution Cluster

FailSafe Tiebreaker Node Example

Figure 5-5 on page 32, assumes the following application failover domains for resource groups RG1-RG4:

- RG1: N1, N3
- RG2: N2, N4
- RG3: N3, N1
- RG4: N4, N2

Figure 5-5 on page 32. shows the following:

- Time 1: The nodes in the cluster are all part of the FailSafe membership.
- Time 2: A router dies and the heartbeat/control network is effectively split in two, leaving two sets of nodes that can communicate heartbeat messages (set 1 is N1/N2; set 2 is N3/N4)
- Time 3: Because nodes N1/N2 are the set containing the tiebreaker, they will both try to reset the nodes in the other set (N3/N4) (assuming that this is how the nodes were defined for reset purposes). The `crsd` daemon will filter out duplicate requests. N1 is able to reset N3 before the reset timeout takes place.
- Time 4: After the reset is successful and the membership of N1/N2 is confirmed, the resource group RG3 on the reset node N3 will be be moved according to its failover policy to N1. Assuming N4 was also successfully reset, its resource group RG4 will similarly move to N2. If N4 does not get reset, it will continue to serve the resources in RG4, but will be in `UNKNOWN` state in the database in the membership formed by N1/N2.

Once N3 and N4 are running again after the reset, they will try to form a membership because they will be unaware that N1/N2 are already in a membership. However, they will not be able to form a membership because they will not have a >50% quorum.

Note: The figure really shows one sequence of events; the different time designations are for illustration purposes.

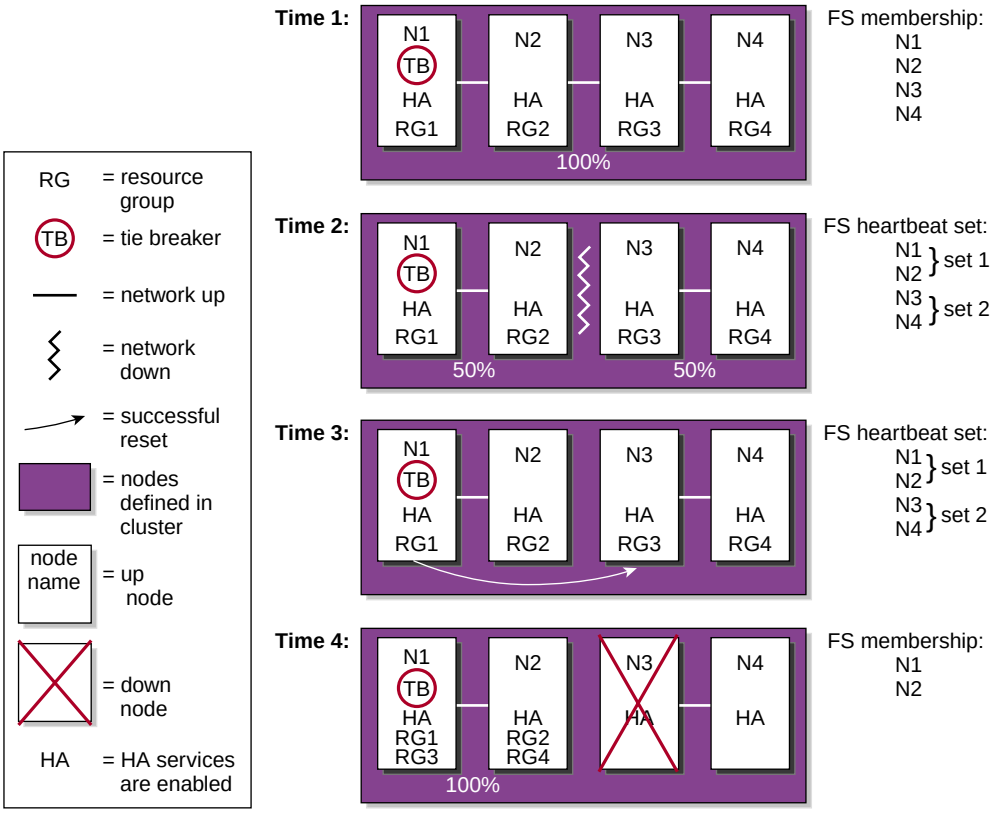

Figure 5-5 FailSafe Tiebreaker Node

Cases when the FailSafe Tiebreaker is Not in the Membership

Just because a group of FailSafe nodes contains the FailSafe tiebreaker, that does not guarantee that it will form or maintain a FailSafe membership. If one group of nodes has the quorum (>50% of the eligible members), it will form the membership whether or not the group contains the tiebreaker node.

In the case of two sets of nodes, each with 50% of the eligible members, the group of nodes that contains the tiebreaker will have the first opportunity to form the membership; however, if the group with the tiebreaker is unable to reset a node in the other group before the reset timeout expires, then the other group will have the

opportunity to reset a node in the first group, and if successful will form a membership even though it does not have the tiebreaker node.

Figure 5-6, assumes the same application failover domains as Figure 5-5 on page 32. Figure 5-6, shows the following:

- Time 1: The nodes in the cluster are all part of the FailSafe membership.
- Time 2: A router dies and the heartbeat/control network is effectively split in two, leaving two sets of nodes that can communicate heartbeat messages (set 1 is N1/N2; set 2 is N3/N4)
- Time 3: Because nodes N1/N2 are the set containing the tiebreaker, they will both try to reset the nodes in the other set (N3/N4), assuming that this is how the nodes were defined for reset purposes. The `crsd` daemon will filter out duplicate requests. However, set 1 is unable to reset a node in set 2 before the reset timeout expires.
- Time 4: Set 2 is now free to try to reset set 1. Both N3 and N4 will try to reset both nodes in set 1; the `crsd` daemon will filter out duplicate requests. N3 is successful in resetting N2.
- Time 5: The membership of N3/N4 is confirmed, the resource group RG2 on the reset node N2 will be be moved according to its failover policy to N4. Assuming N1 was also successfully rest, its resource group RG1 will similarly move to N3. If N1 does not get reset, it will continue to serve the resources in RG1, but will be in `UNKNOWN` state in the database in the membership formed by N3/N4.

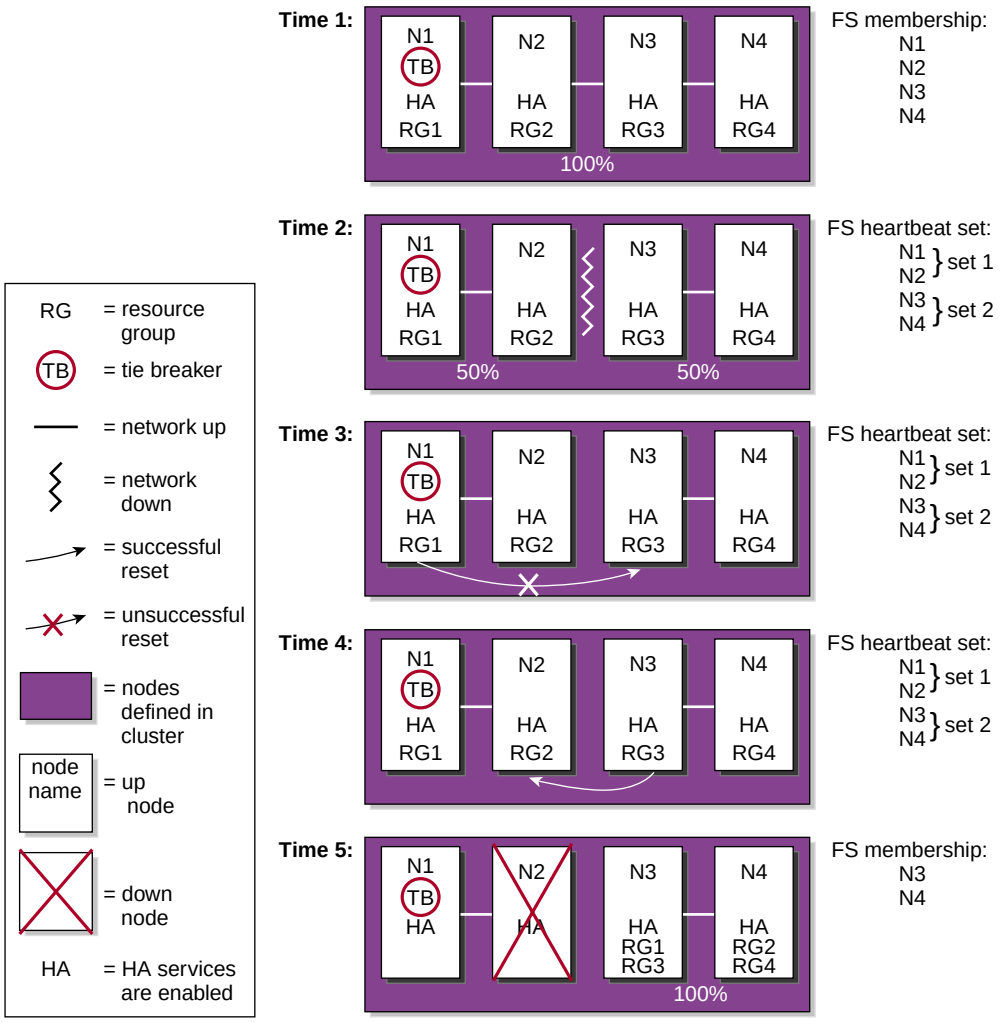

Figure 5-6 Forming a Membership without the FailSafe Tiebreaker

Heartbeat Considerations

There are different heartbeats for each membership type, and each uses a different networking method. Therefore, certain network misconfiguration can cause one heartbeat to fail while another succeeds.

At least two networks should be designated as FailSafe heartbeat networks. FailSafe uses only the highest priority working network for heartbeats; the other network is for heartbeat failover. Usually the private network is used as the highest priority heartbeat network.

In a coexecution cluster, there must be two networks as required by FailSafe; at least one private network is recommended for FailSafe and a private network is required by CXFS.

In a coexecution cluster, CXFS metadata, CXFS heartbeat, and FailSafe heartbeat can use the same network. The heartbeat intervals and timeouts should be appropriately adjusted, if possible, so that all network traffic has sufficient bandwidth. You cannot change the heartbeat timeout or interval for the cluster database membership. Before you adjust the heartbeat settings for the FailSafe membership or CXFS kernel membership, you should consider the impact on the other heartbeats.

If the highest priority network fails, the FailSafe and cluster database memberships will continue using just the next priority network, but the CXFS kernel membership will fail.

Execution of FailSafe Action and Failover Scripts

The order of execution is as follows:

1. FailSafe starts up by using the `start ha_services` command in `cmgr` or as part of the node bootup procedure. It then reads the resource group information from the cluster database.
2. FailSafe tells the system resource manager (SRM) to run `exclusive` scripts for all resource groups that are in the `Online ready` state.
3. SRM returns one of the following states for each resource group:
   - `running`
   - `partially running`
   - `not running`
4. If a resource group has a state of `not running` in a node where HA services have been started, the following occurs:
   a. FailSafe runs the failover policy script associated with the resource group. The failover policy scripts takes the list of nodes that are capable of running the resource group (the *failover domain*) as a parameter.
   b. The failover policy script returns an ordered list of nodes in descending order of priority (the *run-time failover domain*) where the resource group can be placed.
   c. FailSafe sends a request to SRM to move the resource group to the first node in the run-time failover domain.
   d. SRM executes the `start` action script for all resources in the resource group:
      - If the `start` script fails, the resource group is marked `online` on that node with following error:

        ```
        srmd executable error
        ```

      - If the `start` script is successful, SRM automatically starts monitoring those resources. After the specified start monitoring time passes, SRM executes the `monitor` action script for the resource in the resource group.

5. If the state of the resource group has a status of `running` or `partially running` on only one node in the cluster, FailSafe runs the associated failover policy script:
   - If the highest priority node is the same node where the resource group is `partially running` or `running`, the resource group is made online on the same node. In the `partially running` case, FailSafe tells SRM to execute `start` scripts for all resources in the resource group.
   - If the highest priority node is another node in the cluster, FailSafe tells SRM to execute `stop` action scripts for resources in the resource group on other nodes. FailSafe then makes the resource group online in the highest priority node in the cluster.
6. If the state of the resource group is `running` or `partially running` in multiple nodes in the cluster, the resource group is marked with an `error exclusivity` error. These resource groups will require operator intervention to become online in the cluster.

Figure 6-1 shows the message paths for action scripts and failover policy scripts.

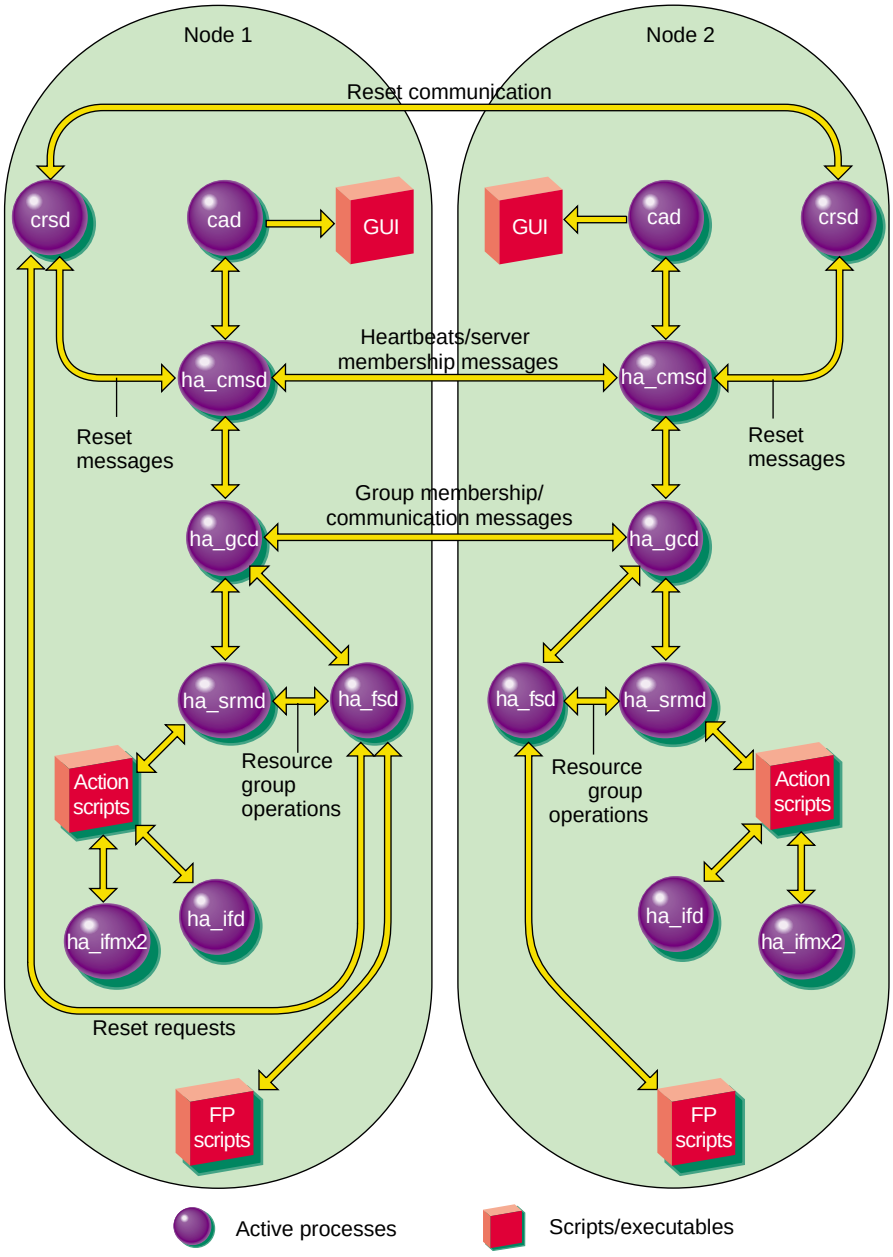

Figure 6-1 Message Paths for Action Scripts and Failover Policy Scripts

When a `start` Script Fails

When the `start` action script fails, the order of execution is as follows:

1. SRM notifies FailSafe of the `start` action script failure as a resource group failure.
2. FailSafe runs the failover policy script to determine the next node for the resource group.
3. FailSafe sends a request to SRM to release the resource group and allocate the resource group in the next node in the cluster.

When a `stop` Script Fails

When the `stop` action script fails, the order of execution is as follows:

1. SRM notifies FailSafe of the `stop` action script failure as a resource group failure.
2. FailSafe marks the resource group with the following error:

   ```
   srmd executable error
   ```

3. The system administrator must use the `offline force` command to clear the error state after stopping the resource group in the node.

Interface Agent Daemon (IFD)

The IFD is an agent that monitors network interfaces and IP addresses. The IFD monitors all network interfaces and IP addresses configured in the node even when there are no highly available IP addresses in the node.

The IFD checks the number of input packets for each interace. If the number of input packets does not increase for a 10–second period, the IFD contacts the broadcast address of the interface by using the `ping` command. If the input packet count does not increase in the next 10–second period, the network interface and all IP addresses on the interface are marked as bad.

The IFD reads the configuration of IP addresses from the cluster database.

`IP_address` resource type action scripts use the `ha_ifdadmin` command to communicate with the IFD. Action scripts obtain status and configuration IP address from the IFD.

IFD logging can be controlled with the GUI and the `cmgr` command.

Index